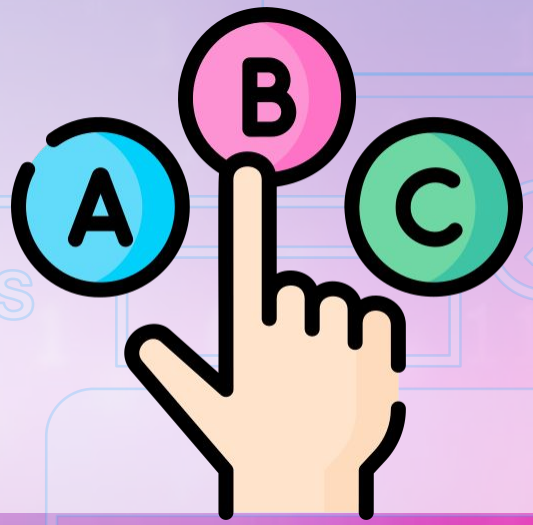
Sentinel

JavaScript Loops

DEFENDING OUR DIGITAL WAY OF LIFE

Recap

We can allow our code to make decisions depending on the inputs that are given to the program



Recap

If... | If... Else... | True, False | <, >, ==, !=

```
if(day == "Monday"){
    promptElement.innerHTML = "Working Day!"
} else {
    promptElement.innerHTML = "Rest Day!"
}
```

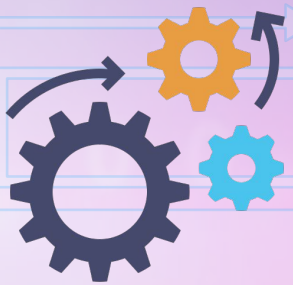
Recap

We can code more complex decisions using logical operators

AND (&&)

OR (||)

NOT (!)



Recap

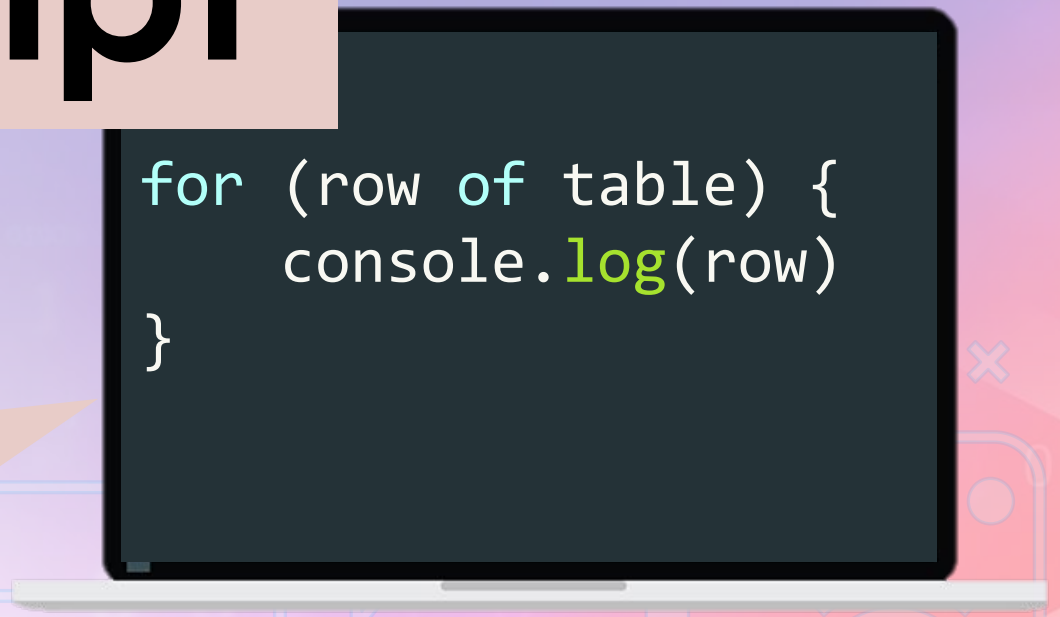
Logical Operator Truth Tables

A	B	A && B	A || B	!(A && B)	!(A || B)
True	True	True	True	False	False
True	False	False	True	True	False
False	True	False	True	True	False
False	False	False	False	True	True

Learning Objectives

What are Loops
When to use Loops
How to write Loops to simplify code

JavaScript



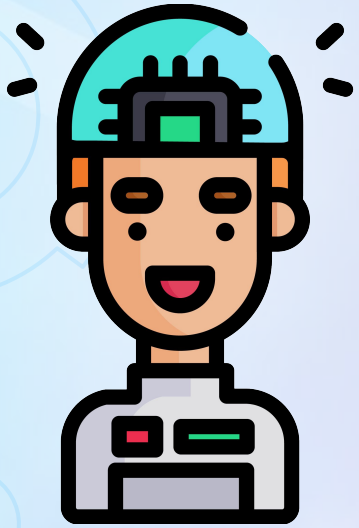
Loops

Exercise Time

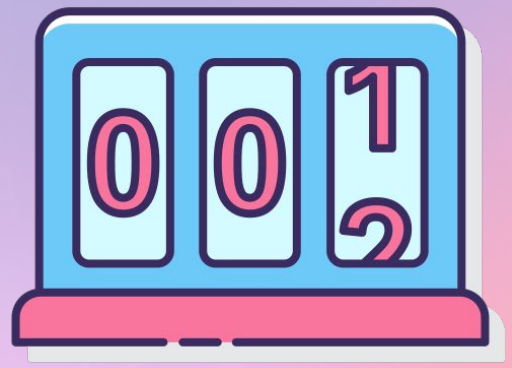
- While you cannot touch the whiteboard, take one step towards it. Make sure you count out loud the steps taken!

- For each step you take, clap your hands that many times

Problem

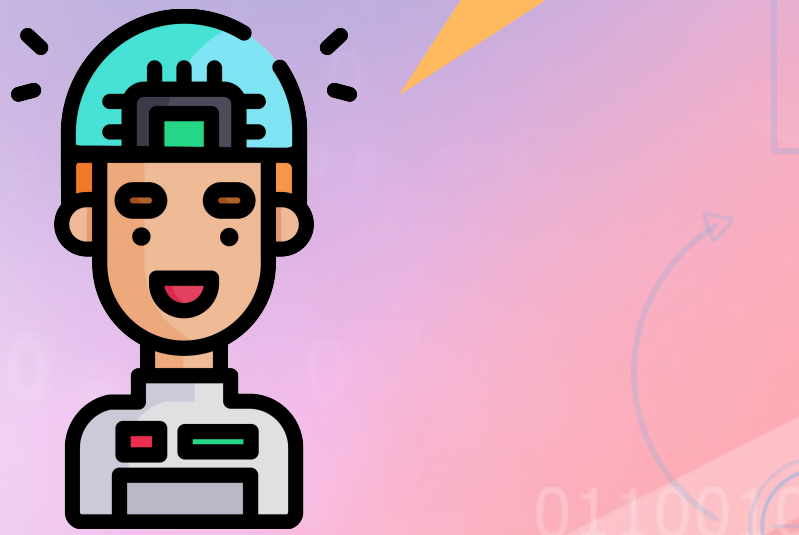


I want to write code that counts to 10.



Solution

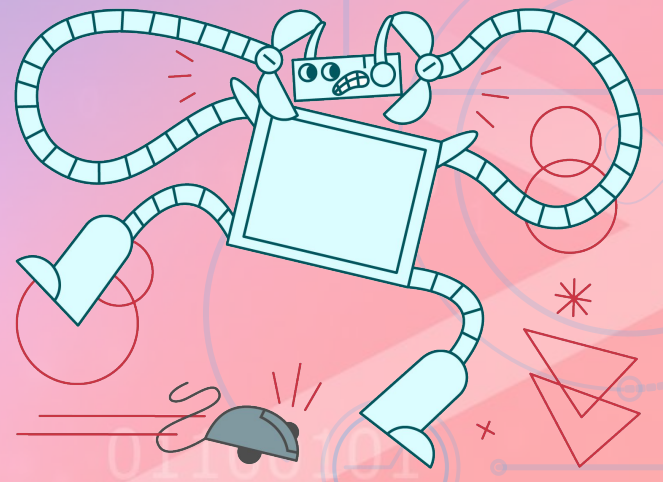
```
console.log(1)
console.log(2)
console.log(3)
...
console.log(10)
```



Problem

I want to write code that counts to **100000**. ☹️

To generalize, in many cases we need to perform an operation multiple times



The real solution: Loops

Loops will execute a block of code multiple times

```
as long as the number <= 100000:
    print the number
    increment the number by one
```

While Loops

While loops will execute a block of code as long as the condition is met

```
let number = 1
while (number <= 100000) {
    console.log(number)
    number++
}
```

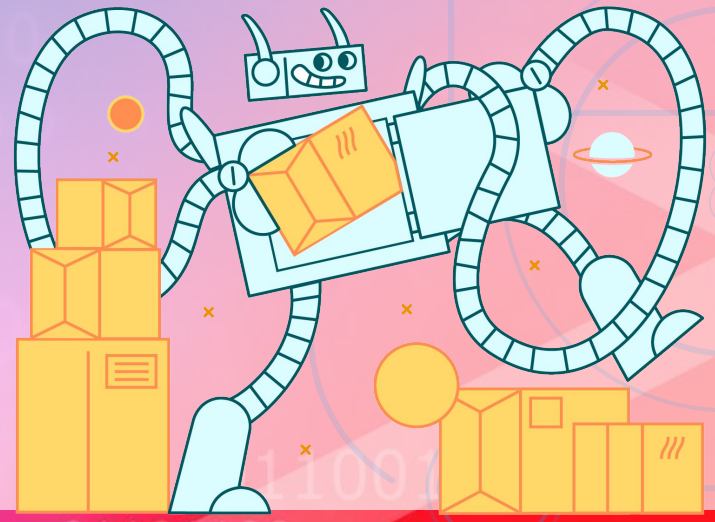
condition

For Loops

Like while loops, but simplify the code

Let's say we wanted to calculate n! (n factorial)

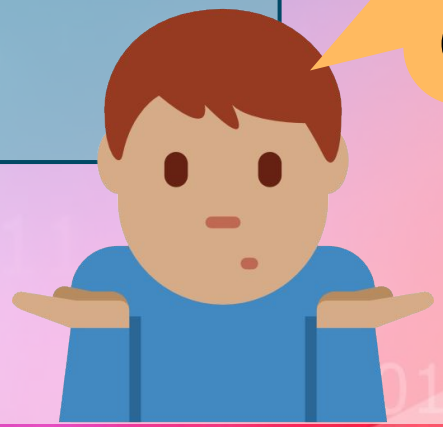
$$1 * 2 * 3 * \ldots * n$$



n!

The algorithm would look something like:

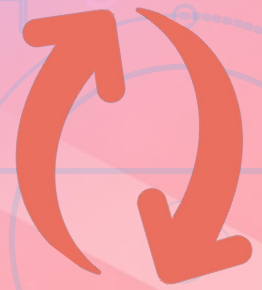
```
let result = 1
for every number from 1 to n:
    result = result * number
```



For Loops

In code this would look like this:

```
let result = 1

for (let number = 1; number <= N; number++) {
    result = result * number
}
```



For Loop

specifies the beginning of a loop

```
let result = 1

for (let number = 1; number <= N; number++) {
    result = result * number
}
```

Initialize the “counting” variable – this happens only once, before the first iteration

For Loop

Loop condition – this condition will be checked before each iteration of the loop

```
let result = 1

for (let number = 1; number <= N; number++) {
    result = result * number
}
```

If the condition is no longer met, the loop breaks and execution continues to the following code

For Loop

This statement is executed every time after the code block has been executed

```
let result = 1

for (let number = 1; number <= N; number++) {
    result = result * number
}
```

For Loop

```
let result = 1

for (let number = 1; number <= N; number++) {
    result = result * number
}
```

Any code in the block will be executed at every iteration of the loop!

Let's write it together

I want to print every multiple of 5 between 0-100
5, 10, 15, ...

And count how many of them are also divisible by 2
10, 20, ...

Solution

```
let count = 0

for (let number = 0; number <= 100; number += 5) {
    console.log(number)

    if ((number % 2) == 0) {
        count++
    }
}
```

Using Loops for Iteration

Most of the time we'll be using loops to **iterate** over data structures

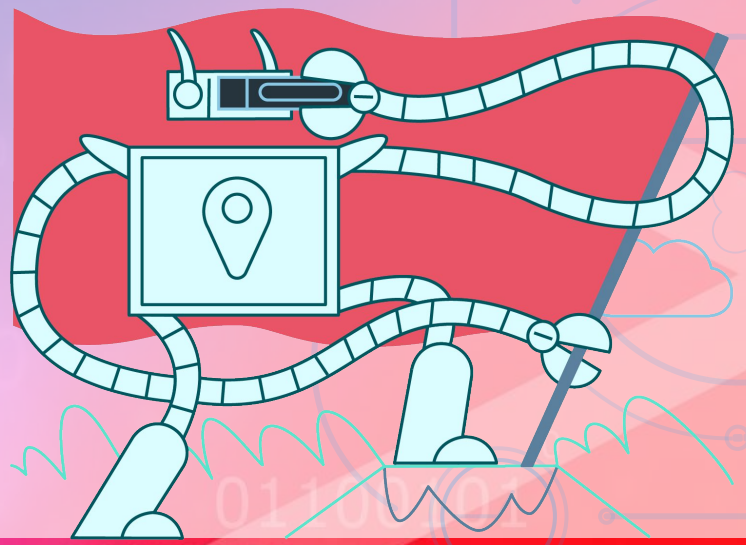
As an analogy, imagine getting a bag of candy

```
For each piece of candy:
  Unwrap the candy
  Eat the candy!
```

Using Loops for Iteration

A practical example is that we need to count the number of occurrences of each letter in our ciphertext to perform frequency analysis for breaking the cipher

But we'll get to that later on…



Iterating over Strings

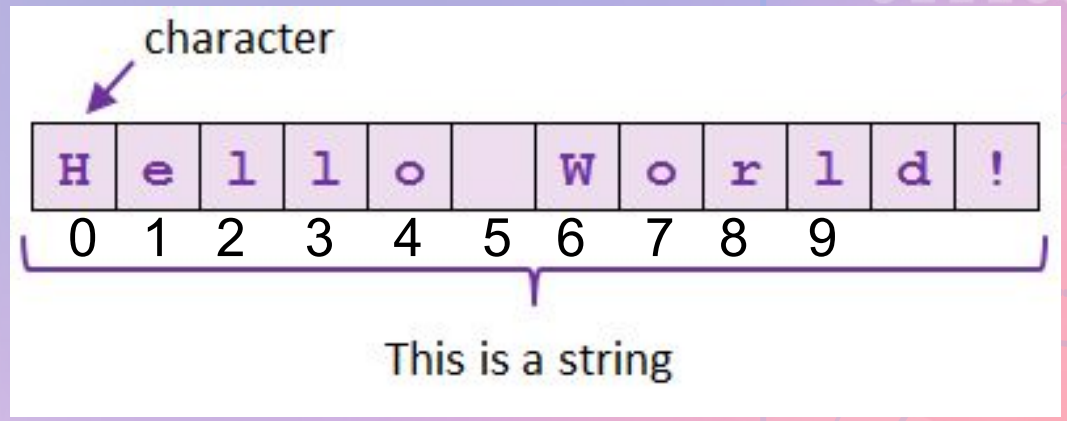
For now, let's try to print out each character in a string

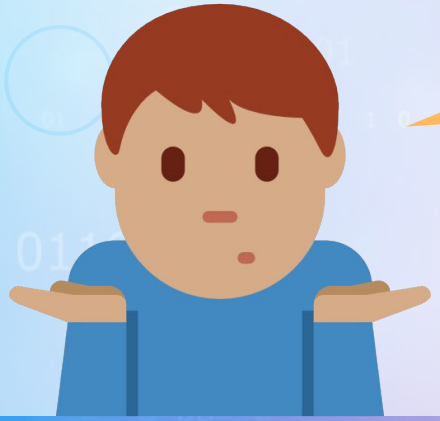
```
let message = "Hear ye!"

for (let index = 0; index < message.length; index++) {
    console.log(message[index])
}
```

Iterating over Strings

Remember:

- Indexing starts at 0!
- index **<** message.length



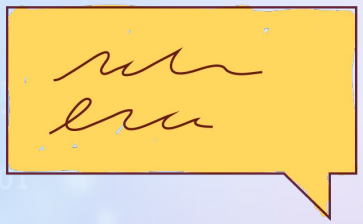


Why not <=?

for … of

To simplify iteration over structures, JavaScript introduced the for … of loop!

The data structure that we're iterating the items of



```
let message = "Hear ye!"

for (const character of message) {
    console.log(character)
}
```

for … of

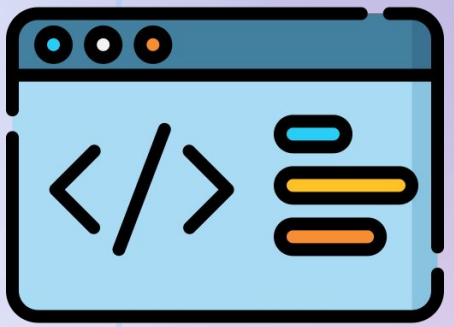
To simplify iteration over structures, JavaScript introduced the for … of loop!

Variable that will be assigned at each iteration of the loop to the next element of the data

```
let message = "Hear ye!"

for (const character of message) {
    console.log(character)
}
```

I Want to Break Free!

What if we need to stop a loop midway?



E.g. write a loop that checks if a string contains a certain character

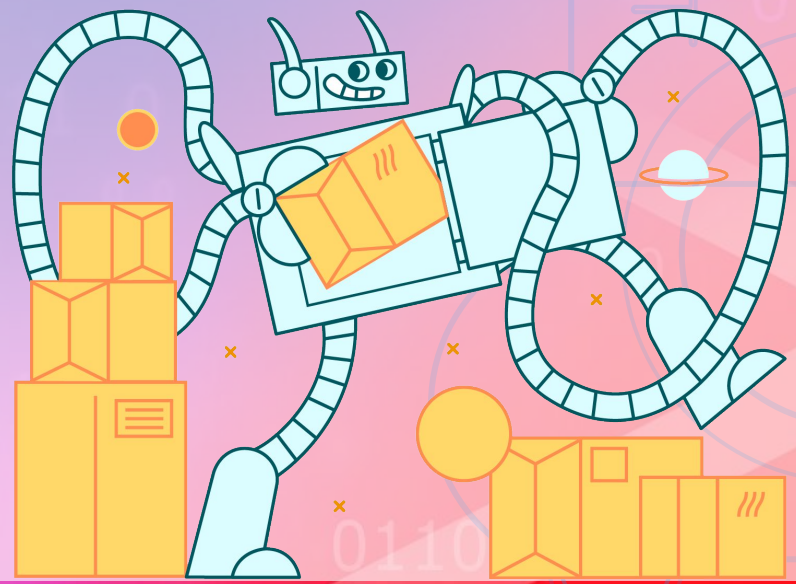
I Want to Break Free!

```
let message = "I want to break free!"

for (const character of message) {
    if (character == "t") {
        console.log("found!")
        break
    }
}
```

Summary

When do we use loops?

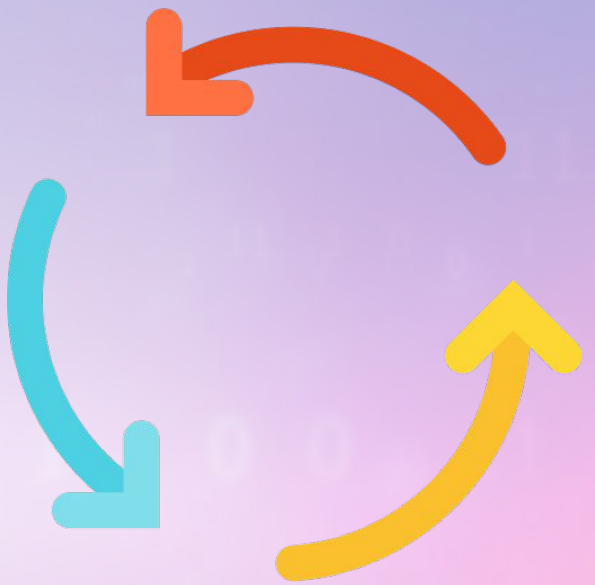
- Loops are used when we want to execute a block of code repeatedly



Summary

What are the type of loops JavaScript has?

For Loops



While Loops

For ... of Loops

Summary

Write a while loop:

```
let number = 0
while (number < 5) {
    /*do something*/
    number++
}
```

Write a for loop:

```
for (let number = 1; number <= N; number++){
    /*do something*/

}
```

Summary

Write a for...of loop:

```
let message = "Hear ye!"

for (const character of message) {
    console.log(character)
}
```

Additional Questions

1. How would you write an infinite loop?

2. Does the following loop include the number 55?

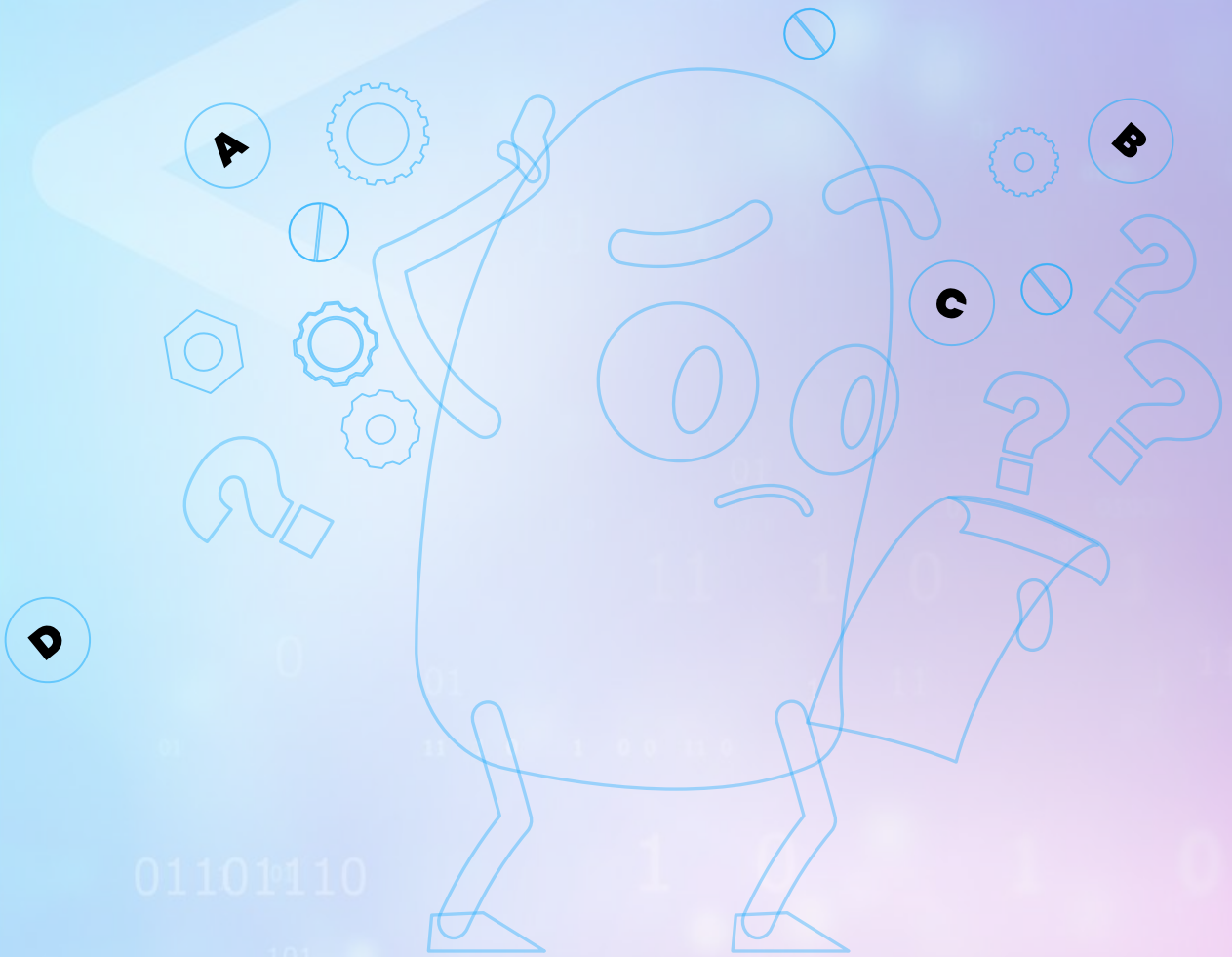
```
for (let i = 0; i < 55; i++)
```

Additional Questions

3 Does the following loop include the number 10?

```
for (let i = 10; i >= 0; i--)
```

4 Does it include the number 0?



Questions?

Sentinel

Your Turn!

> Play around, have fun, ask questions!